0. Prerequisites: things you are expected to know already

- Proof by induction
- Proof by contradiction
- Proof by transposition
- Proof by example/construction
- Recurrences

Proof by induction

mathematical induction is a method used typically to prove a statement for all positive integers, which works in two steps

1. base case: the first step of proving the statement for a given (typically the first) integer (e.g. show the claim is true for $n = 1$)
2. induction case: the second step of proving that the statement for an integer implies the statement for the next integer (suppose the claim is true for $n = k - 1$ and prove it is also true for $n = k$)

example 1. summation

Show $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ for every positive integer n.

a **(formal) proof** by induction is

- Let n be an arbitrary positive integer.
- **base case**: if $n = 1$, then $\sum_{i=1}^{n=1} i = 1$.
- **induction hypothesis:** assume inductively that $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$ for every positive integer $k < n$.
- **induction case:**

$$\begin{aligned}
\sum_{i=1}^{n} i &= n + \sum_{i=1}^{n-1} i \\
&= n + \frac{n(n-1)}{2} && \text{[induction hypothesis with } k = n-1\text{]} \\
&= \frac{n(n+1)}{2} .
\end{aligned}$$

- We conclude that $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

example 2. trees

a **tree** is an undirected graph of n nodes that is connected with no cycle

a **subtree** of a tree T is a connected subgraph of T

a **proper** subtree is any tree except T itself

show that

> The number of edges in a tree with n nodes is $n - 1$.

a formal proof by induction is

- ★ Let T be an arbitrary tree.
- ★ **base case:** if T has one node, then it has no edges
- ★ **induction hypothesis:** assume that in any proper subtree of T with number of nodes at most n, the number of vertices is one more than the number of edges.
- ★ **induction case:** consider an arbitrary tree T with $n + 1$ nodes. Since by assumption a tree is connected with no cycle, if we remove a leaf node i and the edge connecting i to the rest of the tree, the remaining graph is a tree with n nodes. Then, we know from the induction assumption that this tree has $n - 1$ edges. This proves that the original graph with $n + 1$ nodes had n edges.
- ★ We conclude the number of edges is one less than the number of nodes.

often, there exists alternative methods to prove the same claim

for example,

> **a proof (without induction)**: For an arbitrary tree T, choose an arbitrary node to be the root. Because T is connected, there exists a path from the root to every node. Because there is no cycle, there the path is unique. Hence we can define a directed graph by directing every edge of T outward from the root. On this directed acyclic graph, every node has at most one edge directed into it since there is no cycle. Every node except for the root has at least one edge directed into it, since T is connected. We conclude that the number of edges is one less than the number of nodes.

example 3. recursive functions

define a function $F : \mathbb{Z}_+ \to \mathbb{Z}_+$ recursively by setting $F(0) = 0$ and $F(n) = 1 + F(\lfloor n/2 \rfloor)$ for every positive integer n

For every positive integer n, $F(n) = 1 + \lfloor \log_2 n \rfloor$.

a proof by induction:

- Let n be an arbitrary positive integer.
- **base case:** for $n = 1$, $F(1) = 1 + F(0) = 1$.
- **induction hypothesis:** suppose $F(k) = 1 + \lfloor \log_2 k \rfloor$ for every positive $k < n$.
- **induction case:** we separate the proof into two cases, when n is even and when it is odd. Suppose n is an even number:

$$\begin{aligned} F(n) &= 1 + F(\lfloor n/2 \rfloor) \\ &= 1 + 1 + \lfloor \log_2(\lfloor n/2 \rfloor) \rfloor \qquad \text{[by induction hypothesis]} \\ &= 2 + \lfloor \log_2(n/2) \rfloor \;=\; 2 + \lfloor \log_2(n) - 1 \rfloor \;=\; 1 + \lfloor \log_2(n) \rfloor \,. \end{aligned}$$

  when it is odd, $F(n) = 2 + \lfloor \log_2(\lfloor n/2 \rfloor) \rfloor = 2 + \lfloor \log_2((n-1)/2) \rfloor = 1 + \lfloor \log_2(n-1) \rfloor = 1 + \lfloor \log_2(n) \rfloor$, where we used the fact that $\lfloor \log_2(n-1) \rfloor = \lfloor \log_2(n) \rfloor$ for an odd n.
- We conclude that $F(n) = 1 + \lfloor \log_2 n \rfloor$.

Proof by contradiction

to prove a statement to be true, prove that the statement being false implies a contradiction

example: $\sqrt{2}$ is irrational

proof by contradiction:

Suppose $\sqrt{2}$ is rational. Then there exists two integers p and q such that $\sqrt{2} = p/q$ and at least one of which is odd. It follows that $p^2 = 2q^2$, which implies that p is even and can be represented as $p = 2r$ for some positive integer r. It again follows that $q^2 = 2r^2$, which implies q is even.

So both p and q are even, and one of them must be odd, which is a contradiction.

Therefore, the assumption that $\sqrt{2}$ is rational must be false.

example: minimum spanning tree

- ★ given a connected undirected graph, a **spanning tree** is a subgraph that is a tree can connects all the nodes together
- ★ a single graph can have many spanning trees
- ★ given a weighted graph, a **minimum spanning tree** is a spanning tree with weight less than or equal to any other spanning tree

a **cut** (S, S^c) of a graph is a partition of a graph into two disjoint subsets of nodes S and S^c that are joined by at least one edge

a **cut-set** (S, S^c) is a subset of edges whose end points are in different subsets of the partition

if e is an edge in a cut-set (S, S^c) with weight strictly smaller than any other edges in the cut-set, then e is included in any minimum spanning tree

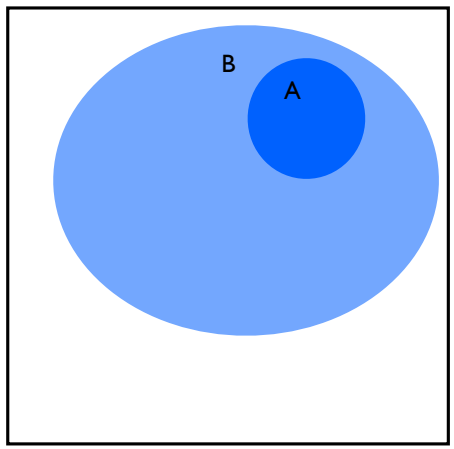
proof by contradiction:

Assume e is an edge with minimum weight in a cut-set (S, S^c). Suppose T is a minimum spanning tree and e is not included in T. Consider T and add the edge e to get a graph G' with a single cycle. Since this cycle includes nodes from both S and S^c, there are at least two edges in G' that connect S and S^c. One of them is e and the other one we call e'. Remove e' from G' and call it T', we know it is going to be a tree since we removed an edge from a cycle in G'. Comparing T and T' the only difference is the edge $e \in T'$ and $e' \in T$. By assumption, we know that $w_e < w_{e'}$. We can conclude that T' has smaller weight than T, and this contradicts our assumption that T is a MST.

Proof by contraposition

to prove "if A then B", show "if not B then not A"

similarly, to prove that $x \in A$ implies $x \in B$ one can show $x \notin B$ implies $x \notin A$



example: for all integer x, if x^2 is even then x is even

example: if a graph $G = (V, E)$ has no cycle and $|E| = |V| - 1$, then G is connected

proof by contraposition:

Assume G has no cycle and G is not connected, then we want to show that $|E| < |V| - 1$.
Let k be the number of disconnected components in G, then we can add $k - 1$ edges to make G connected but ensuring that it still has no cycle. This connected graph with no cycle has $|E| + k - 1$ edges, and from previous lectures, we know that a connected graph with no cycle has number of edges equal to $n - 1$. Hence

$$|E| + k - 1 = n - 1 \, .$$

Since $k \geq 2$ by the assumption that G is disconnected, this implies that $|E| < n - 1$, which finishes the proof.

Proof by example/construction

prove by providing a construction or disprove by a counter-example

example: Eulerian cycle

a **cycle** in an undirected graph is a sequence of nodes starting and ending at the same node, with each two consecutive nodes in the sequence adjacent to each other in the graph

an **Eulerian cycle** is a cycle that visits every edge exactly once

For an undirected graph, if all nodes have even degree, then there exists a Eulerian cycle

proof by construction:

Start at any node and move to a neighboring node erasing the edge you traversed as long as this does not make the graph disconnected. When there is only one neighbor, in which case you have no choice but to make the graph disconnected, you are allowed to move to that neighbor. Since all node degree is even, if we enter a node then we are guaranteed that there is a way out of that node. This process only stops when we have successfully eliminated all the edges and we have reached the node that we started from.

Recurrences

a **recurrence** is a recursive description of a function

a recurrence consists of two types of cases: the base case and the recursive case

$$\begin{aligned} f(0) &= 0 \\ f(n) &= f(n-1)+1 \end{aligned}$$

a function is a solution to a recurrence if it satisfies all statements in the recurrence

$$f(n) = n$$

in many cases, the running time of an algorithm is expressed as a recurrence, or counting the number of objects in a set can also be expressed as a recurrence

goal is to find an exact **closed-form solution** to such recurrences, which gives a non-recursive description of the solution

when exact solution does not exist, or is too complicated, we resort to asymptotic solutions of the form $O(n \log n)$ to $\Omega(n^2/\log n)$

Big-O notation

the big O notation describes the asymptotic behavior of a function, when an argument tends towards a particular value of infinity (e.g. $n \to \infty$)

$$f(n) = O(g(n)) \text{ as } n \to \infty$$

if and only if there exists a positive constant C and a positive integer N such that for all sufficiently large enough $n \geq N$

$$|f(n)| \leq C\,|g(n)|$$

similarly,

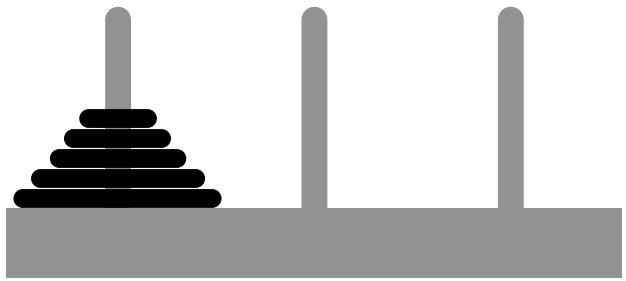
$$f(n) = \Omega(g(n)) \quad \text{iff} \quad f(n) \geq C\,g(n)$$
$$f(n) = \Theta(g(n)) \quad \text{iff} \quad C_1\,g(n) \leq f(n) \leq C_2\,g(n)$$

Guess-and-check

the ultimate fail-safe method for solving a recurrence is guessing the solution and checking the correctness

Tower of Hanoi

there are three rods and n disks of differing sizes



obeying the following rules, the goal is to move a stack of disks sorted in ascending order of the size on one rod to another rod using only a small number of allowed moves

1. only one disk can be moved
2. each move consists of taking the top disk from one stack and placing it on the top of another stack
3. no disk may be placed on the top of a smaller stack

Guess-and-check

(recursive implementation of the) optimal algorithm

assume we have the optimal algorithm for moving $(n-1)$ disks to another rod, then the optimal algorithm is

1. first move $(n-1)$ disks on one rod(using an optimal algorithm for $n-1$ disks, freeing the third rod
2. move the largest disk to the empty rod
3. move the $(n-1)$ disks on the top of the largest rod

the optimality follows from the fact that without moving the largest disk, it is impossible to move all disks to another rod

the proposed algorithm moves the largest disk in a minimum number of moves, hence no other algorithm can achieve the goal with smaller number of moves

the total number of moves $T(n)$ for n disks gives a recurrence:

$$T(n) = 2\,T(n-1) + 1$$

Guess-and-check

expanding the recursion,

$$\begin{aligned} T(n) &= 2\,T(n-1)+1 \\ &= 2^2\,T(n-2)+2+1 \\ &\vdots \\ &= 2^k\,T(n-k)+2^k-1 \end{aligned}$$

since $T(1)=1$, we can now 'guess' the solution, which is

$$T(n)=2^n-1$$

Guess-and-check

Binary search

given a sorted array of n real numbers, find the relative position of a input real-valued key v in the sorted array

binary search

- ★ compare the input key with the value in the middle
- ★ if the key is smaller, apply binary search recursively to the smaller half of the array
- ★ if the key is larger, apply to the larger half
- ★ repeat until the array has only one number

the total number of comparisons, $T(n)$, is given by a recurrence

$$T(n) = 1 + T(\lceil (n-1)/2 \rceil)$$

in the worst case, and expanding the recursion,

$$\begin{aligned} T(n) &= 1 + T(\lceil (n-1)/2 \rceil) = 1 + T(\lceil (1/2)\lceil (n-1/2) \rceil \rceil) \\ &\leq 1 + 1 + T(\lceil n/4 \rceil) \leq k + T(\lceil n/2^k \rceil) \end{aligned}$$

since $T(1) = 1$, we can 'guess' an upper bound on the solution, which is $T(n) = O(\lceil \log_2 n \rceil)$